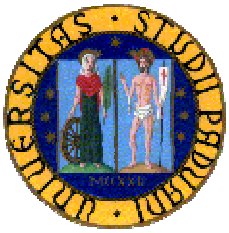


UNIVERSITÀ DEGLI STUDI DI PADOVA

FACOLTÀ DI INGEGNERIA

DIPARTIMENTO DI TECNICA E GESTIONE DEI SISTEMI INDUSTRIALI

TESI DI LAUREA TRIENNALE

CONFRONTO CRITICO FRA CODIFICA IN ANYLOGIC E IN ARENA PER LA SIMULAZIONE DISCRETA.

Relatore: Ch.mo Prof. Giorgio Romanin Jacur

LAUREANDO: Leonardo Bertollo

MATRICOLA: 580873

Anno Accademico 2011-2012

Indice

Introduzione

L’argomento di questa ricerca consiste nello spiegare prima i concetti base dei modelli e della simulazione discreta, e poi applicarli a delle semplici situazioni concrete.
Lo scopo che ci si pone, quindi, è quello di dare al lettore una conoscenza di base di come la simulazione sia utile in tante diverse occasioni: oramai al giorno d’oggi la simulazione sta diventando uno strumento fondamentale anche per studiare e analizzare le situazioni più quotidiane della vita reale.
Ma non solo; attraverso la simulazione è possibile gestire e studiare anche situazioni più complesse di tipo aziendale e riguardanti il mercato, come ad esempio gli ordini di produzione, gli ordini a magazzino per la stima della capacità, i colli di bottiglia, tutto al fine di fare delle prove e migliorare il progetto prima di eseguirlo nel sistema reale.
Creare un modello può essere a volte l’unico modo per studiare un problema, soprattutto se stiamo trattando situazioni molto complesse, o che presentano molti quesiti, o molte variabili, di cui è difficile tenere traccia in modo completo e soddisfacente.

Dopo aver analizzato gli aspetti generali della discussione, ci soffermeremo su due diversi modelli, che riguardano uno un sistema di servizio di una filiale di una banca, e l’altro l’ingresso di una stazione della metropolitana, e li utilizzeremo per fare un confronto tra due diversi programmi di riferimento, AnyLogic e Arena, per valutarne differenze, analogie, difetti, vantaggi e svantaggi.

CAPITOLO 1
Il modello

Un modello è una struttura, un insieme di assunzioni e approssimazioni che vengono prese allo scopo di descrivere e rappresentare il più fedelmente possibile un determinato oggetto, fenomeno o processo reale per fornirne previsioni sul loro stato futuro; esso costituisce quindi un potente mezzo di previsione e descrizione del comportamento di un sistema reale.
Costruire un modello di un sistema significa quindi sostituire al sistema stesso un qualcosa di più semplice e/o più facile da studiare e analizzare, ma riconducibile con una certa fedeltà all'originale nelle sue parti fondamentali.
Il modello è una rappresentazione del fenomeno o del sistema reale in oggetto, e quindi descrive la loro probabile evoluzione e si basa su delle condizioni iniziali forniti dall'utente (dati input, o stimoli); restituisce invece dei dati finali (output).
La creazione di modelli come strumento di aiuto nei processi decisionali è antichissimo e molto diffuso: basti pensare ai modelli in scala, usati soprattutto nella progettazione, che altro non sono che modelli che replicano fedelmente (in scala ridotta) la realtà che si vuole rappresentare.
Esempi di questi modelli possono essere i plastici, utilizzati soprattutto in architettura, oppure i modelli di strutture, usati per studiare gli effetti di sollecitazioni quali ad esempio i terremoti o le frane.
Nonostante queste siano delle azioni diffuse e antiche, oggi è sempre più raro il loro utilizzo a causa dell'elevato costo di produzione e degli elevati tempi di progettazione, senza dimenticare la loro rigidità e l'ingombro che possono avere.
Al giorno d'oggi invece, trovano sempre più diffusione i modelli creati da appositi programmi software su calcolatore.
La corrispondenza tra realtà e modello è di tipo funzionale, cioè ad ogni elemento del sistema reale corrisponde un oggetto informatico (un sottoprogramma, una struttura di dati, ecc.) che ne svolge la funzione nel modello.
Il sistema che si andrà a rappresentare sarà costituito da un insieme di entità che avranno degli attributi e delle proprietà e che interagiscono tra loro per realizzare un obiettivo.
In base a ciò che si vuole simulare, si può ricorrere a diverse tipologie di modelli:

- continui o discreti: la loro differenza sta nel modo in cui il sistema si evolve nel tempo: in un modello continuo lo stato e le variabili cambiano in continuazione,

mentre in un modello discreto le variabili assumono, in precisi istanti di tempo, dei valori ben definiti all'interno di un il sistema che cambia il suo stato solo in corrispondenza di un evento;

- statici o dinamici: un modello statico descrive un sistema in un particolare istante di tempo, mentre un modello dinamico esprime la variabilità e l'evoluzione temporale del comportamento del sistema considerato;
- deterministici o stocastici: la loro differenza la troviamo nella presenza o meno dell'elemento casuale: i modelli deterministici non presentano variabili casuali, quindi l'evolversi del sistema e gli output ottenuti dipendono univocamente dai parametri d'ingresso inseriti nel sistema, mentre i modelli stocastici contengono variabili casuali la cui evoluzione dipende si dagli input, ma anche appunto da elementi casuali inseriti.

1.Le classi di modelli

Ci sono varie tipologie di modelli, che si possono raggruppare in alcune categorie più generali a seconda dell'approccio di modellazione:

- modelli fisici: con modello fisico si intende una rappresentazione concettuale che descrive fenomeni reali e che ne spiega il funzionamento.
  Un esempio classico sono i simulatori di volo;
- modelli in scala: sono una rappresentazione di entità del sistema nella stessa realtà fisica, ma con dimensioni diverse: molte volte le dimensioni del modello sono ridotte, ad esempio i modelli costruiti in architettura o nelle costruzioni navali, ma ci possono essere dei casi anche in cui le dimensioni del modello sono ingrandite: basti pensare ad esempio ad una rappresentazione di un atomo o di una molecola;
- modelli matematici: è un modello costruito usando il linguaggio e gli strumenti della matematica.
  Possono essere modelli analitici, che danno come risultato una formula matematica, oppure simulativi, che riproducono una parte del modello considerato nel modo più completo possibile secondo i dati della simulazione.
  Nei modelli matematici la realtà viene rappresentata secondo alcune variabili logico/matematiche di interesse, per ottenere un modello che raffiguri in modo quanto più veritiero possibile il funzionamento del sistema;
- modelli analogici: sono una rappresentazione delle entità del sistema attraverso una realtà fisica diversa, ma che ne segue le stesse leggi fisiche.

Ad esempio la scienza delle costruzioni oppure i simulatori analogici;

- modelli simbolici: sono modelli che rappresentano la realtà sotto un particolare punto di vista ricorrendo a dei simboli, come ad esempio le carte geografiche o le mappe.

2.Gli elementi presenti in un modello di simulazione discreta

- Entità: le entità sono le parti fondamentali del modello, sono i soggetti del modello, e sono gli elementi che vengono "trattati" dal processo.
  Tali oggetti hanno la caratteristica di essere temporanei, e di subire passivamente le trasformazioni.
  Ad esempio per un'impresa industriale, le entità possono essere considerate le materie prime e i semilavorati, che subiscono le varie operazioni per essere trasformati in prodotti finiti; oppure in una stazione ferroviaria, le entità possono essere i passeggeri che vengono trasportati dalla posizione iniziale alla posizione finale.
  In questo ultimo caso però non parliamo più di processi che riguardano un bene fisico, ma parliamo piuttosto di servizi, le cui entità sono informazioni, documenti o clienti a seconda della necessità e del sistema.
  All'interno del modello, può non interessare tenere traccia del singolo pezzo in lavorazione o in transito nel sistema (per esempio nel caso ci sia un numero elevatissimo di oggetti da considerare), quindi si parla di entità anonime e vengono considerate come un flusso indistinto.
  Al contrario, se si ha interesse a considerare i parametri di lavorazione del singolo pezzo, si parlerà di entità personalizzate.
- Macchine: rappresentano gli elementi fissi del sistema, e i loro stati ne definiscono univocamente la situazione generale.
  Le macchine possono essere fisiche, se sono realmente presenti nel sistema, o logiche, se compiono operazioni fittizie ma presenti logicamente nel sistema (ad esempio nel controllo qualità non ci saranno delle trasformazioni fisiche, ma comunque l'oggetto passa da "oggetto da controllare" a "oggetto controllato").
  L'aspetto fondamentale di una macchina è rappresentato dalle sue prestazioni.
- Attributi: sono le proprietà permanenti di una macchina o di un'entità.
- Stati: sono delle variabili che tengono conto istante per istante, dello stato del sistema e delle sue componenti.
  Possono essere dei numeri oppure dei valori logici.

- Eventi: sono fenomeni che modificano lo stato del sistema.
  Ad esempio l'inizio di una lavorazione in una macchina finora non operativa, modifica lo stato della macchina da libera ad occupata, in modo da far sapere al sistema che in questo istante di tempo quella macchina è già impegnata e non può svolgere altre lavorazioni.
- Operazione: rappresenta la singola trasformazione che avverrà sull'entità.
  Molte volte si parla di cicli di operazioni, e infatti ne possiamo individuare due: il ciclo della macchina, che riguarda gli stati che la macchina attraverserà e le operazioni che eseguirà (quindi in pratica l'insieme di tutte le possibili operazioni che la macchina può compiere, ma anche dei tempi di attesa a cui è soggetta la macchina) e il ciclo del pezzo, che rappresenta il percorso che ciascuna entità fa nel sistema, quindi le macchine attraversate e le operazioni subite.
- Code: sono insiemi di entità che non possono accedere alla trasformazione successiva a causa della non disponibilità momentanea della macchina.
  Bisognerà quindi attendere che la macchina sia di nuovo disponibile per avanzare alla trasformazione successiva.
  Le code più importanti sono: FIFO (first in first out), cioè la prima entità che entra è anche la prima entità che esce; LIFO (last in first out), cioè l'ultima entità che entra è la prima ad uscire; PRI, quando esiste un sistema di selezione degli utenti a priorità; infine c'è anche SIRO (service in random order) , che è il servizio in ordine casuale).
- Cicli di attività: riguardano le relazioni tra le molteplici entità del sistema.
  I cicli di attività corrispondono ad una serie di eventi nei quali ciascuna entità viene associata alla sequenza di più stati che può assumere.
  Lo stato, che può essere attivo, passivo o di stasi, può dipendere anche da più entità, e in uno stesso ciclo i vari tipi di stato si possono alternare.
- Variabili di stato: sono grandezze numeriche o logiche che variano nel tempo perché sono caratteristiche di un certo istante, quindi di un determinato stato del sistema.
- Parametri permanenti: sono sempre grandezze numeriche o logiche, ma al contrario delle precedenti, non variano nel corso della simulazione.
  Questi parametri permanenti sono decisi in fase di modellazione.

3.Le fasi della costruzione di un modello

Il processo per la costruzione di un modello è suddiviso in varie fasi, che devono essere considerate nel giusto ordine per ottenere la riuscita del modello stesso.
È un processo iterativo (conclusa una fase si passa alla successiva), ma ha bisogno di continui feedback: nel procedere si possono riscontrare errori di valutazione o errori logici che costringono a tornare indietro e riesaminare le fasi precedenti.
Le fasi per la costruzione di un modello sono:

1. Formulazione del problema: è necessario, come prima cosa, stabilire cosa si vuole simulare; si deve quindi trovare i confini del sistema che si sta studiando. Questo è importantissimo per focalizzarci su quella parte del processo che ci interessa, ottenendo una simulazione più precisa e utile, e senza correre il rischio di “andare fuori tema”.
2. Scelta degli obiettivi: dopo aver stabilito cosa si vuole simulare, bisogna stabilire gli obiettivi che si intendono perseguire con il modello; bisogna quindi capire quali informazioni si vogliono ottenere dalla simulazione.
   In base allo scopo e agli obiettivi finali, ci saranno delle considerazioni e dei comportamenti diversi e utili alla causa da tenere, come fare delle assunzioni oppure delle semplificazioni di situazioni marginali, che aiuteranno a realizzare un modello significativo per queste imposizioni, ma magari poco utile se utilizzato per scopi o situazioni non simili.
3. Analisi del sistema e definizione del livello di dettaglio: è importante determinare il livello di dettaglio al quale si vuole scendere nel corso dell’analisi del sistema in quanto costruire un modello molto dettagliato è più difficile e dispendioso sia da sviluppare che da correggere nel caso ci siano errori; d’altra parte, semplificare troppo il modello lo rende lontano e riduttivo rispetto alla rappresentazione della realtà.
   Il grado di dettaglio va quindi pesato sempre con il grado di complessità che introduce, considerando però anche la bontà dei risultati che permette di ottenere: è giusto sottolineare che dei modelli troppo semplificativi possono portare a risultati approssimativi o non attendibili.
   Fondamentale è anche considerare il sistema in ogni sua sfaccettatura, ragionando su tutte le sue particolarità ed eccezioni, prima di poter progettare il modello.
   Per fare al meglio questo, molte volte può risultare utile (a volte anche necessario!) immedesimarsi nel sistema e studiarlo e osservarlo da dentro,

facendo proprio finta di essere “un’entità” che subisce le operazioni e le trasformazioni.

4. Raccolta ed elaborazione dei dati: la raccolta dei dati relativi alla realtà da rappresentare, è una fase molto lunga e dispendiosa nella costruzione di un modello, ma fondamentale per riuscire a crearlo bene e funzionante in ogni suo aspetto.
   Successivamente questi dati devono essere analizzati per poter creare le variabili, la sequenza dei processi e la generazione di eventi del modello nel modo più possibile fedele alla realtà.
   Anche questa fase richiede molto tempo, e non è detto non possa essere portata avanti in parallelo con le altre attività.
5. Costruzione del modello: è la fase più delicata e critica dell’intero processo, perché è necessario individuare le entità del modello e le relazioni funzionali che le legano.
   Non è detto esista un unico modo di procedere, ma un approccio soddisfacente anche se empirico potrebbe essere creare il modello base, e poi successivamente andare ad apportare le varie correzioni necessarie delle parti a valle.
   Provando e riprovando a costruire il modello, importando continuamente modifiche e lasciando fuori le parti considerate marginali, si può raggiungere la realizzazione di un modello che rispetti i vincoli principali del sistema ma che allo stesso tempo non trascuri alcuna situazione significativa.
   Si inizia quindi con un livello di dettaglio moderato per poi aumentarlo gradualmente, continuando a confrontare sempre il modello con la realtà.
   Come risultato di questa attività, si otterrà la struttura che verrà poi implementata al calcolatore.
   Creare una documentazione su come è stato sviluppato il modello potrebbe essere una cosa utile sia per facilitare la correzione di eventuali errori, sia per l’utente che poi userà il modello che potrebbe avere problemi nella comprensione.
6. Codifica del modello al calcolatore: una volta creato il modello in forma cartacea, è necessario renderlo interpretabile dal calcolatore, cioè tradurlo in un programma con un linguaggio adatto.
   Per codificare il modello è possibile usare dei linguaggi general purpose come Pascal, C, C++, per i quali esistono delle librerie di routine orientate alla simulazione.

Oppure si può ricorrere ad applicazioni di tipo interattivo per la simulazione, facili da usare e quindi molto adatte a costruire rapidamente modelli (anche sofisticati e complessi), come ad esempio Arena Simulation o AnyLogic.
Per simulazioni di modelli di piccole dimensioni, è possibile utilizzare anche strumenti informatici di uso comune come il foglio elettronico, utile per avere rapidamente un'idea del funzionamento di un singolo componente, o di un sottosistema di un sistema più complesso.
Dopo averlo trasformato nel linguaggio del calcolatore, bisogna verificarlo, cioè controllare che il programma scritto funzioni e che traduca in modo corretto e fedele la logica del modello.
Per la verifica del modello, invece, è preferibile testare il programma a piccoli sottoinsiemi, in modo che sia più facile e veloce identificare e correggere eventuali errori.
Gli errori principali che si possono commettere sono errori di sintassi, i quali basta semplicemente correggerli, oppure errori di concetto, che sono i più difficili da individuare perché sono errori logici che fanno ottenere un risultato differente rispetto alla realtà e quindi all'obiettivo che si vuole raggiungere.

7. Convalida del modello: serve per verificare ci sia corrispondenza tra modello e realtà e, nel caso non ci sia, bisogna capire le cause che comportano gli errori e le differenze.
   Per eseguire questa convalida, bisogna testare il modello con una serie di dati campione, e poi bisogna confrontare i risultati ottenuti: se c'è una buona corrispondenza con un'approssimazione accettabile, si considera valido il modello (ottenere risultati identici è impossibile!).
8. Effettuazione degli esperimenti di simulazione: lo scopo della realizzazione dei modelli di simulazione è poter condurre quanti esperimenti virtuali vogliamo senza dover intaccare il sistema reale.
   Infatti si può osservare il comportamento del sistema in particolari condizioni estreme o di pericolo per la realtà, oppure cosa potrebbe succedere se applicassimo certe modifiche su alcuni parametri.
   Gli esperimenti che possono essere condotti sono principalmente di due tipi: un primo approccio di tipo interattivo, che consiste semplicemente nel far funzionare il modello e osservare cosa succede, magari per vedere come reagisce il sistema a seguito di una modifica, e un secondo approccio di tipo comparativo, nel quale uno stesso parametro viene modificato più volte per vedere come si comporta il sistema nelle varie situazioni, per poi poter scegliere

la soluzione che più si avvicina alla realtà o che meglio ottimizza i parametri del modello.

9. Analisi e presentazione dei risultati e delle conclusioni: i risultati ottenuti dai vari esperimenti vengono raccolti e analizzati attentamente, magari ricorrendo all'utilizzo di tabelle e grafici che ne semplificano la lettura.
   Per ritenere i risultati accurati ed attendibili, può essere necessario far funzionare più volte la simulazione del modello, per poi procedere a fare una media di tutti i dati raccolti.
   In questo modo si sarà certi della bontà del lavoro svolto.

CAPITOLO 2
La simulazione

Simulazione è un termine generico che comprende una serie di metodi e applicazioni che consentono di imitare o rappresentare, tipicamente attraverso un calcolatore, delle operazioni eseguite nel tempo da un sistema o da un processo reale.
Viene applicata spesso a problemi riguardanti le code.
Simulare prevede la costruzione di un modello che deve essere validato, ed è quindi molto utile per effettuare degli esperimenti sul modello stesso, che si evolve nel tempo, al fine di trovare l'alternativa più appropriata tra diverse configurazioni.
La simulazione inoltre, è uno strumento molto utile per individuare i punti critici di un sistema e per trovare l'alternativa più appropriata all'interno di un insieme di configurazioni proposte da un decisore; tuttavia essa non è adatta a contesti in cui si richieda la condizione di ottimo, perchè potrebbero esistere soluzioni migliori non ancora analizzate.
I modelli di simulazione consentono di tener conto delle distribuzioni temporali dei valori delle variabili e di ipotizzare diverse soluzioni senza realizzarle fisicamente, riducendo così i costi di attuazione e i rischi derivanti da una cattiva scelta.
Simulazione è sinonimo di realizzazione di un modello della realtà che consente di valutare e prevedere l'evoluzione dinamica di una serie di eventi, che devono rispondere all'imposizione di certe condizioni poste dall'utente o dall'analista.
A questo fine ci si avvale di un calcolatore, che permette di studiare il comportamento del sistema nelle condizioni presenti (o anche in condizioni modificate), senza però modificarlo nella realtà.
Sin subito dalla sua nascita, la simulazione ha riscosso un grande successo e ha trovato ampie applicazioni in molti settori, dal campo sanitario all'industria, allo scopo di ridurre i costi, migliorare l'efficienza, ma anche aumentare la redditività di queste realtà complesse difficilmente testabili se non in via teorica.
Ad oggi, l'utilizzo della simulazione trova un notevole sviluppo sia nel settore industriale che nel settore dei servizi, dove è diventato uno strumento fondamentale per attività di gestione, sviluppo nuovi prodotti e attività manageriali in generale.
Questo perché i processi modellabili e dalla cui simulazione si può ricavare benefici sono moltissimi: si può pensare ad esempio ai sistemi di trasporto, alla gestione del personale e degli aspetti logistici in generale, ai processi di produzione, alla gestione

delle scorte, all'analisi dell'utilizzazione delle risorse, all'ottimizzazione delle procedure, ai servizi, alla sanità, alle telecomunicazioni, ecc.
La simulazione permette quindi di simulare il modello, riproducendo e imitando la realtà in condizioni di laboratorio, cioè in un ambiente di studio facilmente controllabile e manipolabile: è possibile studiare, misurare, migliorare, progettare e controllare un sistema esistente nella realtà, provandolo virtualmente in assoluta sicurezza e senza paura di creare danni a cose o persone.
Nel settore delle simulazioni, prende una notevole importanza la parte riferita al funzionamento dei processi produttivi e logistici.
Tali processi infatti, sono caratterizzati da un'elevata complessità, dovuta alle numerosissime relazioni tra i diversi sistemi che ne fanno parte, alla stocasticità dei parametri del sistema, e ad eventuali guasti e all'indisponibilità di certe risorse.
Per esempio in questi processi devono essere considerati il tempo per l'arrivo delle materie prime, la durata necessaria per le lavorazioni, il tempo che serve ad ogni operatore per svolgere il suo compito ecc., ma vanno considerati anche possibili imprevisti e il tempo perso a correggere questi imprevisti, come la manutenzione straordinaria.
Quindi una buona simulazione prevede anche una valutazione anticipata di questi effetti e delle loro conseguenze sulla capacità di produzione, sulle scorte ecc., e prevede anche il giusto dimensionamento dei magazzini, delle macchine, degli attrezzi, del personale.
Insomma la simulazione deve tenere conto di tutti i parametri che possono far cambiare il corso degli eventi di un sistema ed è una grande fonte di informazioni utili per il processo che si sta analizzando, perché consente l'analisi della realtà ad un elevato grado di dettaglio riducendone la complessità.

1.Fattori positivi e negativi della simulazione

Come per tutte le cose, anche per la simulazione possiamo trovare sia dei pro che dei contro, anche se i vantaggi sono molto maggiori e più importanti degli svantaggi.
I fattori positivi che derivano dall'uso della simulazione sono:

- Riduzione dei costi: innanzi tutto non può esserci che una riduzione dei costi, perché andare a svolgere degli esperimenti direttamente sul sistema reale potrebbe essere molto oneroso, e a volte magari neanche fattibile.
  La simulazione invece riduce in maniera drastica i costi perché permette sia di individuare eventuali errori prima che vengano commessi nella realtà ed evitare

quindi di dover tornare sui propri passi sprecando denaro ed energia, sia di monitorare e valutare in anticipo le conseguenze dovute alle generiche scelte prese nella gestione del problema e dovute anche all'apporto di eventuali modifiche al sistema;

- Ripetibilità infinita del fenomeno scelto: se si pensa alla realtà, ci si rende conto che a volte non è sempre possibile sperimentare il sistema tantissime volte.
  Utilizzando la simulazione invece, si può ripetere la stessa sequenza di eventi alle stesse condizioni iniziali, o anche partendo da input differenti: questo permette di confrontare direttamente i risultati dello stesso problema per scegliere la situazione migliore, quella ottimale.
  Infine l'esperimento, essendo ripetibile un numero arbitrario di volte, a differenza del caso reale, da la possibilità di raccogliere gli stessi dati tantissime volte;
- Riduzione dei tempi di esecuzione: con la simulazione si possono ottenere gli stessi risultati che si otterrebbero nella realtà riducendo di molto il tempo necessario, grazie al fatto che è possibile comprimere il tempo simulato.
  Inoltre non è detto neanche che si possa concludere un esperimento su un sistema reale a causa del fatto che questo può richiedere tempi lunghissimi di esecuzione.
  Un esempio può essere un esperimento chimico, dove ci potrebbe essere una reazione molto molto lenta, oppure se si vuole fare una previsione sull'evoluzione futura della Terra, dove il tempo necessario sarebbe milioni di anni;
- Riduzione dei rischi: grazie alla simulazione, è possibile testare varie soluzioni, anche quelle che potrebbero essere ritenute dannose o pericolose, o addirittura illegali.
  A volte invece, è proprio necessario simulare virtualmente nuove idee per capire che potrebbero avere conseguenze pericolose per la sicurezza.
  Un esempio potrebbe riguardare i fiumi e la costruzione di dighe, oppure la costruzione di un nuovo ponte, vedere qual è il limite massimo di mezzi che possono transitare contemporaneamente ecc.

Poter studiare e monitorare dinamicamente un sistema senza perturbare la realtà offre quindi l'opportunità di testare tutte le scelte che si possono prendere in assoluta sicurezza, a basso costo e con tempi di esecuzione ragionevoli, senza correre il rischio di compromettere qualcosa o di non potere tornare indietro.
Lo scopo della simulazione è perciò quello di ricreare virtualmente in un ambiente controllato le dinamiche che avvengono nel sistema reale, per mostrare innanzi tutto l'evoluzione del sistema nel tempo, ma anche quali parti del sistema dipendono da quali altre parti, quali sono i parametri che lo influenzano, e perché no, per cercare di evidenziare dei problemi e cercare soluzioni per risolverli.

È quindi evidente l'importanza della simulazione come strumento di supporto per tutte le decisioni, anche quelle banali e quotidiane, non solo quelle che si riferiscono a grandi problemi complessi e difficili da risolvere.

Non ci sono solo note positive sulla simulazione, anche se va detto che i contro sono veramente poca cosa in confronto ai vantaggi.
Alcuni svantaggi possono essere:

- Possibili tempi lunghi per lo sviluppo del modello da simulare: lo sviluppo del modello potrebbe richiedere molto tempo e molta energia spesa; più il modello è complesso e delicato, e più i tempi per avere una simulazione significativa e sufficiente possono essere lunghi.
  Le operazioni di programmazione per ottenere dei dati sufficientemente sensati e tali da dare una corrispondenza tra modello e realtà, possono essere assai lunghe;
- Possibili costi elevati del personale: simulare un modello potrebbe rivelarsi anche oneroso dal punto di vista economico se pensiamo che il personale richiesto per la simulazione debba essere altamente qualificato.
  Per esempio, per l'incorporazione di variabili stocastiche è necessaria un'accurata conoscenza statistica per l'analisi dei risultati in output;
- Approssimazione dei risultati: i risultati ottenuti dalla simulazione sono un'approssimazione della realtà, perciò danno solo un'indicazione di quello che sarà il comportamento del sistema, non danno la certezza assoluta che l'evolversi della situazione sarà così e basta (anche perché poi nella realtà ci sono tantissimi imprevisti che è difficile calcolare), non danno delle risposte esatte;
- Interpretazione dati complessa: per identificare la soluzione migliore, l'output della simulazione deve essere interpretato e questa interpretazione dei dati può rivelarsi un'analisi molto complessa.

2.La simulazione discreta

La simulazione discreta riproduce al calcolatore il comportamento di sistemi quando questo evolve per eventi.
Questi hanno durata nulla e si susseguono in modo discontinuo, segnando il passaggio del sistema da uno stato ad un altro.
Il più piccolo sistema discreto è la fila d'attesa con un servente, in cui sono presenti due soli eventi: l'arrivo di un nuovo cliente e la fine del servizio erogato al vecchio cliente.

La simulazione si basa sulle possibilità di calcolo che offre l'informatica e rappresenta un ottimo strumento valido per lo studio di sistemi dinamici caratterizzati da un'elevata complessità e casualità.
In questo modo è possibile conoscere la dinamica del sistema, variare parametri e condizioni per verificarne la reazione, evidenziare nel modello ciò che è soddisfacente e ciò che è preferibile migliorare.
La corrispondenza tra realtà e modello non è basata su una riduzione in scala delle dimensioni, ma è di tipo funzionale: ad ogni elemento del sistema reale corrisponde un oggetto informatico che svolge la stessa funzione reale ma dentro al modello.

2.1.I sistemi di code

La simulazione discreta sviluppa il modello di un sistema come sequenza di eventi che accadono in istanti discreti nel tempo, ed è particolarmente utile per descrivere sistemi di code.
Questi sono sistemi caratterizzati da un flusso di entità e da un servizio ad esse reso in un determinato tempo.
Per l'erogazione del servizio le entità arrivano e, nel caso questo sia occupato, devono attendere in coda il proprio turno.
In questi sistemi si possono distinguere quattro aspetti fondamentali:
1. La sorgente, che può essere:
    - finita, e in questo caso la coda non potrà che avere lunghezza limitata;
    - infinita, quindi la coda continuerà ad alimentarsi ed eventualmente esplodere.
2. Gli arrivi dei clienti nel sistema, che si possono distinguere in:
    - deterministici;
    - casuali indipendenti;
    - casuali dipendenti dallo stato del sistema;
    - a gruppi;
    - numero massimo di clienti ammessi nel sistema (code a capacità limitata).
3. Le code e le loro regole di priorità, che sono:
    - FIFO (First in first out);
    - LIFO (Last in first out);
    - PRI (esiste un sistema di selezione degli utenti in base alla priorità);
    - SIRO (service in random order, servizio in ordine casuale).
4. Il servizio, che considera quattro elementi:
    - La stazione: luogo fisico dove si compie il servizio;

- Il servizio: attività svolta dal servente sul cliente; è governato da una legge di servizio che può essere deterministica o stocastica;
- Il servente: è colui che compie il servizio; può essere:
  singolo, cioè solo una persona (o risorsa) svolge il servizio; multiplo, cioè più risorse erogano il servizio; oppure vi possono essere infiniti serventi, cioè il numero dei serventi è sempre maggiore del numero dei clienti (non vi è mai coda).
- I tempi in cui viene erogato il servizio; possono essere: deterministici oppure casuali indipendenti oppure ancora casuali non indipendenti.

CAPITOLO 3
Analisi codifica in AnyLogic

AnyLogic è uno strumento software di programmazione per la creazione di modelli di simulazione sviluppato da XJ Technologies.
XJ Technologies è un fornitore leader di strumenti di simulazione dinamica, tecnologie e servizi di consulenza per le applicazioni aziendali.
Tecnologie XJ è stata fondata nel 1992, e ad oggi è un team di 30 professionisti e la loro sede principale si trova in Russia.
AnyLogic è presente anche nel Nord America (la filiale di vendita e di consulenza si trova nel New Jersey) e anche nel resto dell'Europa, precisamente a Parigi.
Grazie a una consolidata rete di 27 distributori in tutto il mondo, AnyLogic è uno degli strumenti più potenti e flessibili per quanto riguarda la simulazione; è la scelta di migliaia di utenti in tutto il mondo, centinaia di organizzazioni commerciali e governative e centinaia di università anche perchè fornisce un grande supporto e una buona consulenza, anche aziendale.
Il loro team di consulenza comprende una combinazione unica di matematica, ottimizzazione dei sistemi, sviluppo software, problematiche di simulazione, quindi riesce ad avere la capacità di comprendere rapidamente gli elementi essenziali del business dei loro clienti e di identificare che cosa e come può essere migliorato, e nel tempo hanno assunto il vantaggio competitivo di avere ottime conoscenze su come funziona la modellazione e la simulazione nelle applicazioni di business.
La comunità di utenti AnyLogic è in costante crescita sia a causa della migrazione da altri strumenti, sia perché AnyLogic riesce a rendere la modellazione e la simulazione applicabile continuamente a nuove aree ed è diventato anche uno standard aziendale per la simulazione in molte aziende globali.
Infatti il linguaggio di AnyLogic ha una flessibilità senza precedenti e consente al modellatore di catturare la complessità e l'eterogeneità dei vari sistemi che possono essere presi in considerazione, e permette di ottenere una conoscenza più approfondita ad ogni livello desiderabile di dettaglio, sui processi su cui si sta lavorando.
All'inizio degli anni '90, il gruppo di ricerca Distributed Computer Network (DCN) della Technical University di San Pietroburgo ha sviluppato un sistema di software per analizzare, verificare e correggere vari programmi: questo sistema permetteva di programmare graficamente la struttura e il comportamento dei sistemi.

Nel 1998 il successo di questo progetto ha ispirato il laboratorio DCN a fondare una nuova società per riuscire a sviluppare un software di simulazione di nuova generazione.
Gli sforzi vennero concentrati nello sviluppo dei metodi quantitativi di: simulazione, analisi delle prestazioni, comportamento dei sistemi aleatori, ottimizzazione e visualizzazione.
Il nuovo software, entrato nel mercato nel 2000, è basato sugli ultimi vantaggi delle tecnologie dell'informatica: un approccio orientato agli oggetti, l'utilizzo del moderno linguaggio di programmazione Java, una moderna interfaccia grafica, ecc.
La prima versione di AnyLogic è AnyLogic 4, poiché la numerazione prosegue la numerazione di COVERS 3.0 (il primo software sviluppato dall'azienda).
Nel 2003 si è fatto un grande passo con il rilascio di AnyLogic 5, incentrato sulla simulazione di business nei seguenti settori:

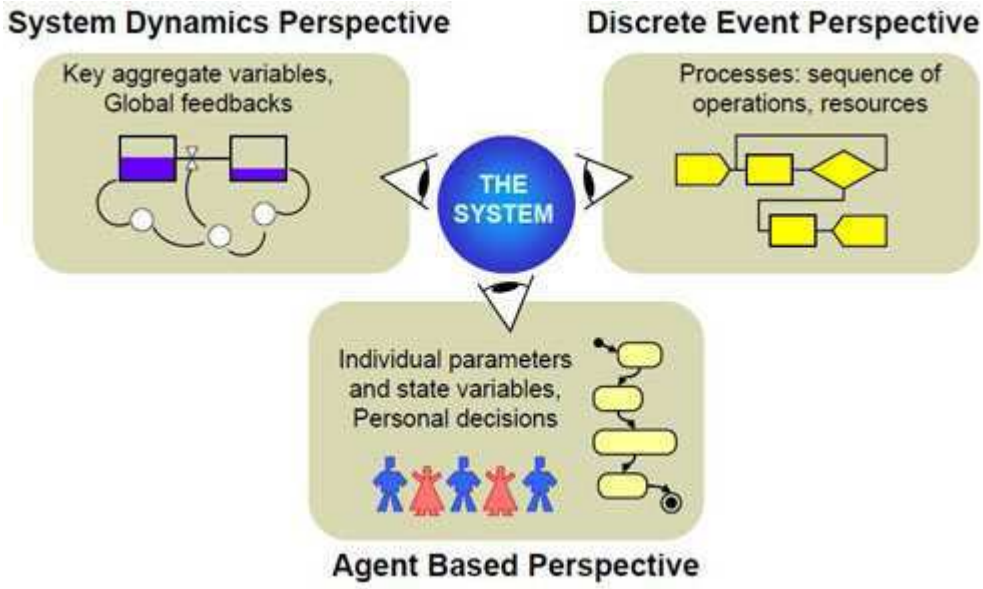
- Mercato e competizione;
- Sanità;
- Produzione;
- Supply Chain;
- Logistica;
- Vendita al dettaglio;
- Processi aziendali;
- Scienze sociali e dinamiche dell'ecosistema;
- Difesa e militare;
- Project and Asset Management;
- Infrastrutture e trasporti;
- Dinamica delle folle e simulazione del traffico cittadino;
- Aerospaziale;
- Fotovoltaico.

L'ultima versione di AnyLogic è la 6, ed è stata rilasciata nel 2007 e la piattaforma su cui si fonda l'ambiente di sviluppo dei modelli è Eclipse.
AnyLogic 6 è un software di simulazione multipiattaforma (cross-platform) in quanto funziona su tutti i più importanti sistemi operativi esistenti: Windows, Mac OS e Linux.

1.Il modello di simulazione multi-metodo

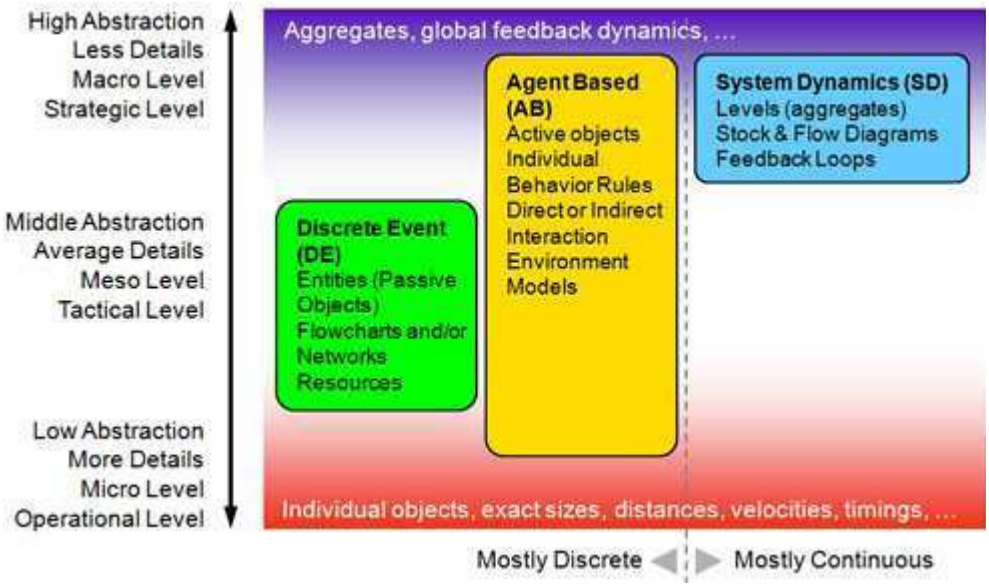
Lo strumento è stato chiamato AnyLogic perché supporta tutti e tre gli approcci di modellazione:

- orientati alle attività: dinamica dei sistemi (System Dynamics);
- orientati ai processi: simulazione ad eventi discreti (Discrete Event Simulation);
- orientati agli agenti (Agent Based Modeling).



È possibile combinare ognuno di questi approcci all'interno di un unico modello.
La system dynamics e il discrete event sono i due approcci tradizionali alla simulazione: l'approccio attraverso la system dynamics è segue i processi continui mentre il discrete event lavora prevalentemente con passo temporale discreto, come per esempio il passaggio da un evento all'altro.
Storicamente le simulazioni system dynamics e discrete event sono state insegnate nelle università a due categorie di studenti molto differenti tra loro, rispettivamente studenti di economia da una parte, e dall'altra ingegneri gestionali e industriali che si occupano di ricerca operativa.
Come risultato oggi ci sono due differenti filosofie di pensiero differenti che non collaborano tra di loro.
Differenti approcci di modellazione corrispondono a diversi livelli di astrazione dei problemi da affrontare.

System dynamics, lavorando con flussi aggregati e tempi continui, è ovviamente usata al più alto livello di astrazione, mentre discrete event è utilizzata ad un livello di astrazione basso e medio basso (processi fisici).



AnyLogic permette all'utente di combinare insieme, nello stesso modello, questi tre approcci secondo l'esigenza, senza nessuna gerarchia fissa da rispettare.
Questo approccio di misto è direttamente applicabile ad una ampia varietà di problemi complessi e che difficilmente possono essere rappresentati attraverso l'utilizzo di un solo approccio se non con grandi compromessi.

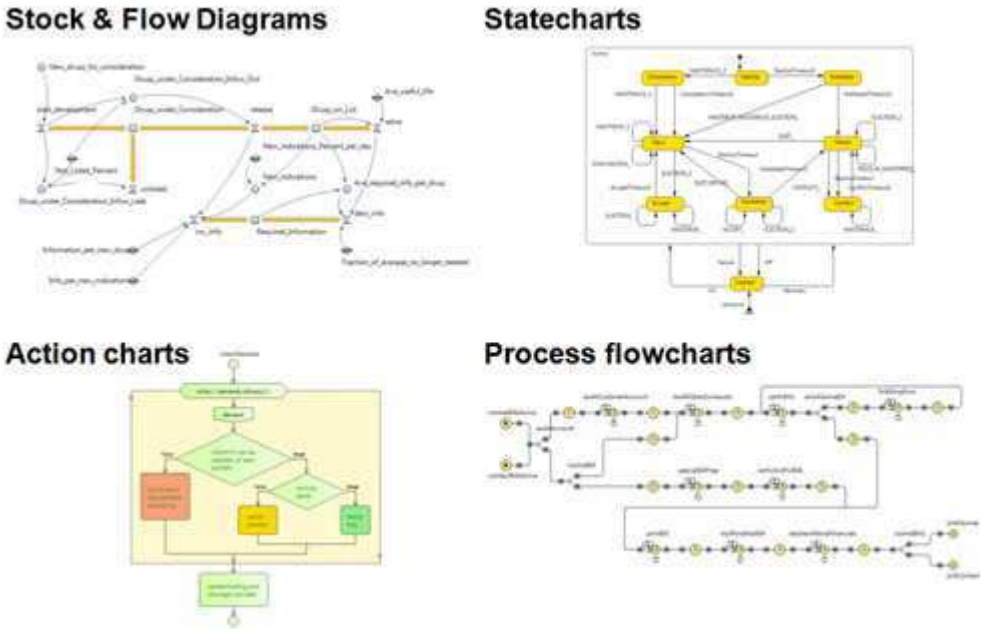
2.Il linguaggio della simulazione

Il linguaggio di simulazione di AnyLogic è formato dai seguenti elementi:

- i diagrammi di scorte e flusso sono utilizzati solo per la modellazione system dynamics;
- i diagrammi di stato (statechart) vengono utilizzati soprattutto nella modellazione agent based per definire il comportamento dell'agente, ma talvolta sono usati anche per simulazioni discrete, come ad esempio per simulare il guasto ad una macchina e quindi il suo stato di attività/fermo;
- i diagrammi di azione vengono utilizzati per definire algoritmi.

Essi possono essere utilizzati nella modellazione discreta, ad esempio nell'instradamento delle chiamate nei call center, o nella modellazione agent based, ad esempio per la logica di decisione dell'agente;

- i diagrammi di flusso di processo, infine, sono la costruzione di base utilizzata per definire un processo negli eventi discreti.



Il linguaggio comprende anche gli elementi base di modellazione a basso livello (variabili, equazioni, parametri, eventi, ecc), le forme di presentazione e gli elementi grafici (linee, polilinee, ellissi, ecc), le strutture di analisi dei risultati (dataset, istogrammi, plot), gli strumenti per connettere fonti esterne di dati, le immagini standard, e la struttura degli esperimenti.

3.Le librerie di AnyLogic

Il linguaggio AnyLogic include tre principali librerie standard:

- Enterprise Library: progettata per supportare la simulazione nelle aree di produzione, supply chain, logistica e sanità.
  Utilizzando gli oggetti della Enterprise Library è possibile modellare sistemi reali in termini di entità (operazioni, transazioni, clienti, prodotti, parti, veicoli, ecc.), di processi (sequenze di operazioni che coinvolgono in genere le code, i ritardi, le attese, l'utilizzo delle risorse), e delle risorse.

I processi sono rappresentati nella forma flowchart, cioè attraverso diagrammi di flusso.

- Pedestrian Library: dedicata a simulare flussi pedonali in un ambiente fisico. Consente di creare modelli di costruzioni ad alta frequentazione di persone (come le stazioni della metropolitana, i controlli di sicurezza, gli aeroporti ecc.) o strade. I modelli riescono a dare una raccolta di statistiche sulla densità di pedoni in zone diverse: questo permette di garantire prestazioni a un livello di servizio accettabile in base a un carico ipotetico di persone, stimare la durata di permanenza o percorrenza di aree specifiche, e rilevare i potenziali problemi causati dalla geometria delle aree interne, come ad esempio l'effetto di aggiungere troppi ostacoli lungo un percorso, e da altre applicazioni.
  In modelli creati con la Pedestrian Library i pedoni vengono gestiti in uno spazio continuo, reagendo a diversi tipi di ostacoli (muri, porte, aree varie), tra i quali anche gli altri pedoni.
  I pedoni sono simulati come attori interattivi dal comportamento molto complesso, anche se la Pedestrian Library di AnyLogic fornisce un'interfaccia di alto livello attraverso il diagramma di flusso per la creazione rapida, semplice e veloce di modelli di gruppi di persone.
- Rail Yard Library: supporta le operazioni di simulazione, modellazione e visualizzazione di tutte quelle operazioni logistico-ferroviarie di qualsiasi complessità, dimensione e scala.
  I modelli di cantiere ferroviario possono essere combinati con modellazione discrete ed agent based legati a: carico e scarico merce, allocazione di risorse, manutenzione macchinari, e altre attività legate al trasporto.

Accanto a queste tre librerie generiche, ciascun utente può creare e sviluppare le proprie librerie e può anche distribuirle.

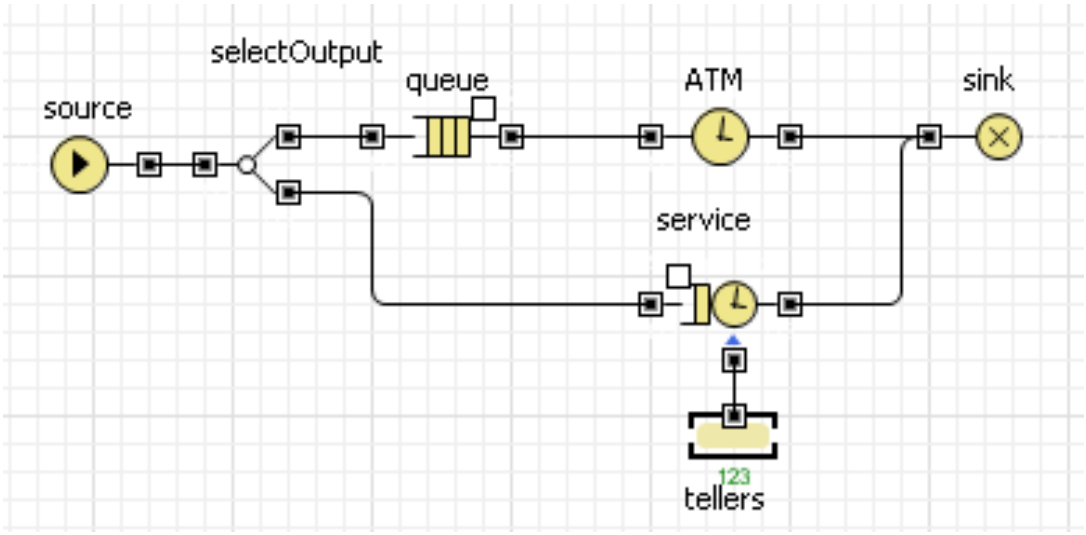
4.L’interfaccia grafica di AnyLogic

AnyLogic è un ambiente di modellazione virtuale di sistemi discreti, continui ed ibridi. Con questo strumento è possibile creare prototipi di sistemi durante le fasi di studio, progettazione o sviluppo, attraverso cui esplorare aspetti e dettagli della progettazione o della implementazione dei relativi sistemi in modo semplice e privo di rischi.
AnyLogic comprende un linguaggio di modellazione grafica e permette anche all'utente di estendere i propri modelli di simulazione con il codice Java.
La natura Java di AnyLogic si presta ad estensioni del modello personalizzato tramite il codice Java, in particolare alla creazione di Java applets che possono essere aperti con qualsiasi browser tramite l’utilizzo della Java virtual machine (JVM).
Questo è molto importante e utile perché rende molto facile condividere modelli creati in AnyLogic attraverso semplici siti web.
In aggiunta agli applets Java, la versione Professional permette anche la creazione di applicazioni Java runtime, che possono essere distribuite agli utenti; queste applicazioni Java sono considerate una base per lo strumento di supporto alle decisioni.
L’interfaccia grafica di AnyLogic, i suoi strumenti e gli oggetti libreria consentono di accedere molto velocemente al modello di diverse aree come la produzione e logistica, processi aziendali, risorse umane, dei consumatori e/o il comportamento di un paziente.
Il modello di progettazione orientato ad oggetti supportato da AnyLogic prevede la costruzione modulare, gerarchica, e incrementale di modelli di grandi dimensioni.
AnyLogic è basato sull’ innovativa struttura di supporto Eclipse (che è stato adottato anche da diverse aziende leader come una piattaforma di applicazioni di business), che permette ad AnyLogic di funzionare su tutti i più importanti sistemi operativi esistenti (Windows, Mac, Linux, ecc) con lo stesso look e feel originale.
Un grande progetto può essere suddiviso in componenti che sono sviluppate da modellisti diversi, e poi unito senza problemi: è possibile addirittura simulare l'intera popolazione di una grande città con ogni persona che abbia proprietà individuali.
L’interfaccia utente grafica che interagisce con l'animazione è una parte importante di AnyLogic.
L'editor di animazione è una parte dell’ ambiente di sviluppo del modello e supporta una grande varietà di forme grafiche, e controlli per la progettazione di interfaccia (slider, pulsanti, input di testo, ecc), così come l'importazione di immagini e file CAD come elementi e sfondi.
AnyLogic comprende una vasta gamma di analisi dei dati e oggetti di grafica

professionale, come grafici a barre, grafici a torta, grafici temporali, e istogrammi.
Questi sono concepiti in modo da rendere efficienti i processi e visualizzare dinamicamente i cambiamenti di dati durante l'esecuzione della simulazione.
AnyLogic impiega Java come linguaggio per la definizione della struttura complessa di dati, algoritmi, e la connettività esterna.
Se necessario, il programmatore può estendere le funzionalità dei costrutti grafici di AnyLogic con pezzi di codice Java, che offre una flessibilità praticamente illimitata.
Ovunque si sta digitando, AnyLogic suggerisce quali variabili o funzioni possono essere utilizzate in un contesto che elimina errori di battitura, nonché la necessità di fare riferimento ad altre parti del modello o un riferimento Java.
Java rende i modelli di AnyLogic cross-platform (accessibili da tantissime piattaforme software) e li rende anche pubblicabili come applets, quindi gli da la possibilità di essere eseguiti a distanza in un browser web.
I modelli sono organizzati con una struttura ad albero, in modo tale da consentirne una facile navigazione al loro interno.
Il Modello è una descrizione di un problema definito in termini di linguaggio di modellazione di AnyLogic.
Ogni modello viene creato singolarmente e rappresenta un insieme di oggetti attivi, che rappresentano gli oggetti del mondo reale, e di esperimenti, che definiscono le opzioni di lancio del modello.
Il modello è l’elemento al più alto livello nell’area di lavoro e viene visualizzato nella finestra Project.
Una volta che un modello di simulazione è completo, un programmatore in genere lo usa per definire ed eseguire vari esperimenti.
AnyLogic supporta molti tipi di esperimenti, che si possono combinare tra loro per creare esperienze personalizzate.

5.Primo esempio: filiale di una banca

Il primo esempio di modello considerato è un semplice sistema di servizio di una filiale di una banca, costituita da uno sportello automatico (Bancomat/ATM), che offre agli utenti un servizio rapido di self-service per il prelevamento di contanti, e da dei cassieri, che compiono invece operazioni più complesse, come ad esempio il pagamento delle bollette.
L'obbiettivo è quello di valutare, a seconda del tasso di arrivo dei clienti, il numero necessario di sportelli automatici e di cassieri affinché le rispettive code non si congestionino.



Elementi fondamentali del modello:

- Source: è la parte che genera i clienti con una specifica frequenza; è la fonte di partenza del modello;
- SelectOutput: è un blocco di decisione a doppia via, che divide i clienti tra chi deve andare al bancomat e chi ha bisogno dei cassieri;
- Queue: è la rappresentazione dei clienti in coda in attesa di essere serviti;
- Delay (ATM): è il tempo di operazione del bancomat;
- Service: è il luogo che gestisce il servizio dei cassieri;
- Tellers: rappresentano i cassieri;
- Sink: rappresenta il completamento del servizio e quindi la fine del modello.

Gli input del modello sono:

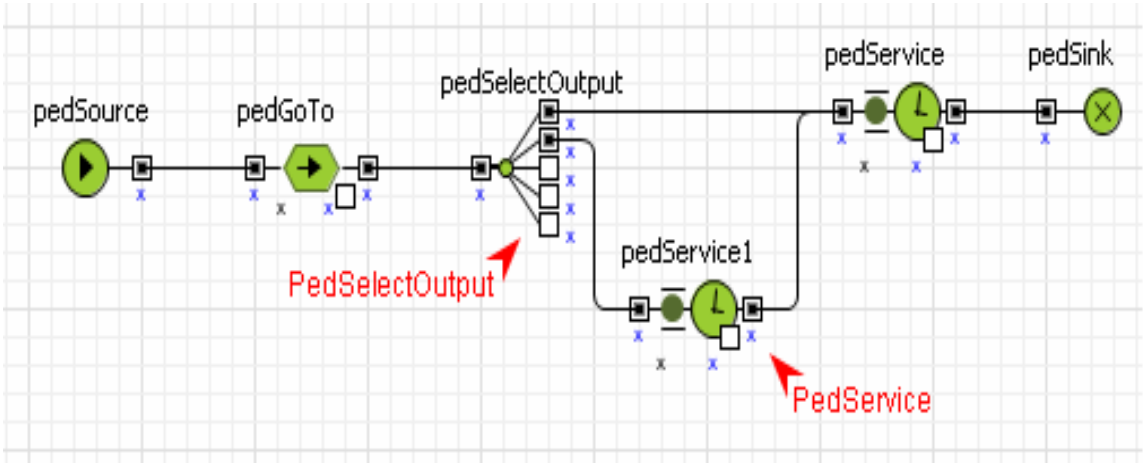
- La frequenza d'arrivo dei clienti è di 0,67 clienti/minuto con una distribuzione costante; per semplicità si è previsto che arrivi un solo cliente contemporaneamente;
- I clienti si dividono equamente tra bancomat e cassieri: 50% bancomat - 50% casse;
- La capacità massima del bancomat è di 15 clienti;
- La capacità massima delle casse è di 20 clienti;
- Il tempo medio che ogni cliente spende allo sportello del bancomat è di 1 minuto;
- Il tempo medio che ogni cliente spende alle casse è di 6 minuti.

Le conclusioni del modello sono:

- Il bancomat non rivela problemi di congestione delle code, quindi un bancomat è più che sufficiente per soddisfare il fabbisogno dei clienti senza farli aspettare troppo in coda;
- Il numero ottimale di cassieri è variabile tra 2 e 3, a seconda dell'affluenza dei clienti.

6.Secondo esempio: stazione di una metropolitana

Il secondo esempio analizza invece l'ingresso di persone in una stazione della metropolitana, creando un modello semplice che simuli il flusso di pedoni.
Il fine è quello di stimare l'affluenza massima che può essere gestita affinché l'ingresso non si congestioni, ed eventualmente studiare delle possibili soluzioni alternative.



Il modello è creato anche in questo caso da un diagramma di flusso che descrive il processo.
Il modello di base simula un flusso di passeggeri dall'entrata in metropolitana, fino all'uscita, dividendo i pedoni in due gruppi in base al fatto che abbiano già acquistato il biglietto oppure che debbano ancora acquistarlo.
Elementi fondamentali del modello:

- pedSource: è la parte che genera l'arrivo del flusso dei pedoni con una specifica frequenza; è il punto di partenza del modello;
- perGoTo: questo è un blocco che muove i pedoni dalla posizione attuale verso una nuova, così si vedono i passeggeri spostarsi dall'ingresso verso i treni della metropolitana.
  È una parte che serve per collegare il diagramma di flusso alla parte grafica;
- pedSelectOutput: è un blocco di decisione a multi-via (in questo caso ne servono solo due), che divide i pedoni in due gruppi: quelli che hanno già il biglietto e quindi possono accedere ai cancelli automatici, e quelli che devono acquistare il biglietto presso la biglietteria.
  Questo è necessario quindi per indirizzare i passeggeri non muniti di biglietto alle biglietterie e gli altri direttamente ai cancelli, quindi serve per poter prendere una

decisione: il pedone che arriva all'oggetto pedSelectOutput è inoltrato verso una delle due porte di output in base ai rapporti specificati per ciascuna di queste porte;

- pedService: esso rappresenta uno o più servizi e ne definisce le proprietà.
  In questo caso definisce il servizio dei cancelli automatici per i pedoni che hanno già acquistato il biglietto, e che quindi non devono fare la coda allo sportello della biglietteria ma solo recarsi allo spazio apposito per timbrare il biglietto e poi prendere la metropolitana;
- pedService1: definisce i parametri di un servizio che rappresenta gli sportelli della biglietteria della metropolitana, quindi definisce il servizio attraverso cui passeranno i pedoni.
  I pedoni dopo aver acquistato il biglietto, si potranno dirigere come gli altri ai cancelli automatici per prendere la metropolitana.
- pedSink: esso nasconde i pedoni in arrivo, per questo è generalmente utilizzato come punto di fine del flusso di pedoni e quindi punto finale del modello.

Gli input del modello sono:

- La frequenza d'arrivo dei pedoni prevede un affluenza massima di 5000 persone/ora;
- Le biglietterie sono 2, mentre i cancelli automatici sono 4;
- La probabilità con cui i pedoni si dividono tra chi ha già il biglietto e chi no non è equa: 85% tornelli, 15% biglietterie.
  In questo modello si è ipotizzato che il numero di passeggeri che sono già in possesso del biglietto sia molto maggiore rispetto al numero di quelli sprovvisti;
- Il tempo medio che ogni pedone spende alle biglietterie è di 25 secondi.

Le conclusioni del modello sono:

- L'affluenza ottimale dei pedoni è stimata tra le 1500 e le 2500 persone/ora;
- Aumentando la frequenza d'arrivo dei pedoni si riscontrano gravi problemi in prossimità delle biglietterie.
  Si può intervenire incrementando il numero di biglietterie, migliorando in questo modo l'efficienza della stazione.

CAPITOLO 4
Traduzione in Arena

Arena è un potente software di simulazione di eventi e software di automazione, e permette quindi di creare modelli ed eseguirvi le necessarie simulazioni.
Fu inizialmente sviluppato da Systems Modeling e successivamente fu acquisito dalla Rockwell Automation nel 2000.
La Rockwell Automation è una multinazionale che si occupa di automazione industriale, controllo e soluzioni informatiche, ed è una delle più grandi aziende nel settore: ha circa 19000 dipendenti in oltre 80 Paesi.
In Arena l'utente costruisce un modello che viene descritto da un diagramma di flusso attraverso l'inserimento dei moduli, che sono delle scatole di forme diverse che rappresentano ognuna dei processi o eventi logici diversi e che devono essere posizionate secondo una precisa disposizione.
Delle linee di connessione sono utilizzate per unire questi moduli insieme e specificare quindi il flusso delle entità.
La creazione di un modello e della relativa simulazione può richiedere molto tempo all'inizio, ma se il tutto viene ottimizzato, alla fine i tempi complessivi necessari per la riuscita del progetto vengono ridotti.
Arena è interamente basato su un proprio linguaggio incorporato denominato SIMAN (Simulation Modeling Analysis) grazie al quale non è necessario scrivere le righe di codice perché l'intero processo di creazione del modello di simulazione è grafico, visivo e integrato.
Arena si integra molto bene con le tecnologie Microsoft.
Esso comprende il linguaggio Visual Basic in modo da poter ulteriormente automatizzare, se necessario, gli algoritmi specifici dei modelli.
Supporta inoltre l'importazione di diagrammi di flusso esterni, così come fogli elettronici Excel o database di Access.
Supporta anche il controllo ActiveX.
Per questo è utilizzato da molte grandi aziende impegnate nella simulazione di processi di business, come ad esempio la General Motors, IBM, Nike, Xerox, Lufthansa ecc.
Arena è in grado di simulare diversi tipi di operazioni, tra cui call-center per l'ottimizzazione dell'uso di agenti e delle linee telefoniche, le dimensioni e

l’instradamento degli oggetti in un impianto di trasformazione di prodotti alimentari; è in grado di gestire servizi clienti, produzione, assistenza sanitaria ecc.
Con Arena è possibile modellare i processi, simulare future performance del sistema per capire le complesse relazioni e identificare eventuali opportunità di miglioramento, visualizzare le operazioni con animazione grafica dinamica, analizzare come il sistema si evolve nel tempo in modo da poter scegliere la soluzione migliore per gestire il progetto.
È un software utilizzato per simulazioni di scenari reali complessi e trova ampia applicazione soprattutto nell’ambito della gestione aziendale.

1.Le componenti del linguaggio Arena

Il software in dotazione che si utilizzerà è la modalità “Student” di Arena versione 13.5.
E’ quindi possibile utilizzarlo in dimensione limitata in termini di numero di moduli e di entità.
Per quanto riguarda il software completo di tutte le funzionalità, la società proprietaria del software, mette a disposizione varie versioni di Arena, acquistabili esclusivamente on-line tramite il sito web.
Il linguaggio Arena si basa su alcuni componenti base fondamentali, che servono per strutturare un modello:

- Le entità: sono oggetti che si spostano all’interno del sistema durante la simulazione variando i valori e lo stato del sistema stesso; rappresentano gli oggetti dinamici della simulazione, come ad esempio clienti, pezzi, lotti, veicoli, ma anche informazioni, elementi logici, ecc.
  Alla fine della simulazione, queste entità sono destinate a lasciare il sistema.
  Tutte le entità devono essere definite e la loro presenza è fondamentale;
- Le code: sono aree di attesa dove il movimento delle entità è temporaneamente sospeso, per un periodo più o meno lungo in base alle caratteristiche del modello; può succedere che un’entità non può occupare un’unità di risorse perché occupata da un'altra entità, quindi l’entità necessita di un posto dove attendere e questo posto viene denominato coda;
- Le risorse: sono componenti del sistema che devono essere allocati alle entità, cioè un’entità occupa una risorsa durante l’operazione in corso per poi liberarla quando ha terminato di sfruttare i suoi servizi.

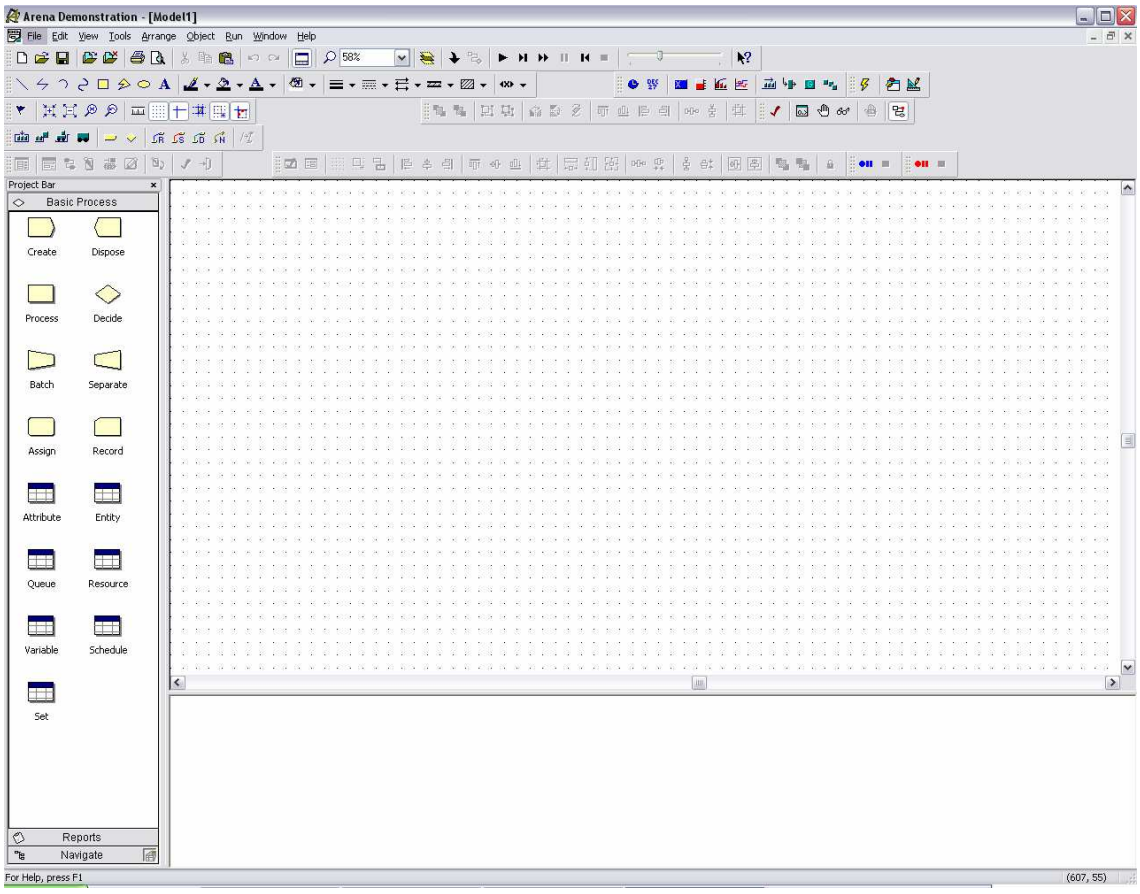
È la risorsa ad essere ceduta all'entità e non viceversa poiché un'entità può utilizzare più risorse.

Alcuni esempi di risorse sono le macchine, gli operatori, i robot, ecc.;

- Gli attributi: sono dei valori che vengono associati alle singole entità; possono riguardare tempistiche (tempo di arrivo, tempo di attesa, tempo di lavorazione), il tipo di lavorazione, quantità (quantità massima), ecc.;
- Le variabili globali: rappresentano dei valori, delle informazioni che descrivono lo stato del processo o del sistema, come il numero di macchine disponibili, il numero di setup necessari, ecc.;
- Gli eventi: un evento è un qualcosa che può succedere in un determinato istante e che può cambiare lo stato del sistema.

  Sono eventi ad esempio l'entrata di un nuovo pezzo nel sistema, e anche la sua uscita, oppure l'arrivo di un nuovo cliente.

2.La costruzione di un modello in Arena



Questa è la pagina iniziale appena si avvia il programma.
In alto sono presenti tutti i tasti comuni ma necessari, come il tasto per salvare il modello, il tasto per aprire un modello già esistente ecc.
A destra invece sono disponibili i moduli fondamentali per la costruzione del modello, moduli che vedremo poi in dettaglio successivamente.
La parte centrale è la parte dove viene creato e sviluppato il modello, mentre lo spazio bianco sotto è la zona dove vengono evidenziate le varie proprietà degli oggetti che si inseriranno nel modello.
Per la costruzione del modello in Arena, quindi, si può ricorrere a due tipologie distinte di moduli:

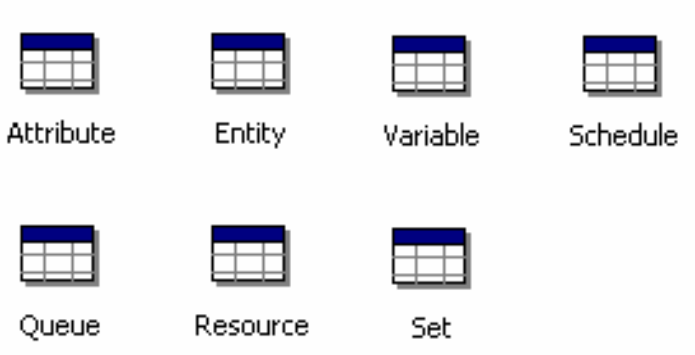
- Moduli flow-chart: vengono inseriti nella finestra del modello e collegati tra loro per formare un diagramma di flusso che serve per descrivere il processo dinamicamente; rappresentano nodi e stadi attraverso cui le entità fluiscono e tipicamente sono connessi tra loro.
  I collegamenti permettono di definire la corretta sequenza dei moduli attraverso i quali le entità fluiscono.

- Create: definisce le modalità di creazione di un'entità, potendo settare il tempo che deve intercorrere tra due arrivi successivi; è inteso come il punto di partenza del modello per le entità;
- Dispose: è inteso come il punto finale del modello per le entità; è la distruzione vera e propria dell'entità; le statistiche dell'entità distrutta possono essere raccolte prima della sua distruzione;
- Assign: viene utilizzato per assegnare dei nuovi valori a variabili e attributi;
- Process: è il modulo che serve per processare un'entità; rappresenta l'attività che l'entità sta svolgendo; modella un ritardo che coinvolge le

entità correnti; consente di simulare acquisizione e rilascio di una o più risorse da parte dell'entità stessa.
Può essere di quattro tipi: delay, dove viene richiesto un tempo di processamento ma senza impiego di risorse; seize delay (rilascio successivo), dove è richiesto un tempo di processamento e anche una risorsa che però viene allocata ma non rilasciata; seize delay release, dove le entità occupano alcune unità di una risorsa (spesso dopo aver atteso in coda), vengono ritardate del tempo di processo, poi rilasciano le unità di risorsa precedentemente allocata; delay release indica che una o più risorse sono già state assegnate e che l'entità semplicemente ritarderà e rilascerà la risorsa specificata;

- Decide: è un processo di tipo decisionale, e il processo di decisione implementato è basato su una condizione o su una probabilità; a loro volta le condizioni possono essere basate su valori di attributi, di variabili, espressioni, tipo di entità ecc.;
- Batch: segue un processo di raggruppamento di entità, che può essere temporaneo oppure permanente; da la possibilità di specificare il numero di entità in ingresso necessarie per creare un nuovo raggruppamento;
- Separate: è un processo che da la possibilità di separare una singola entità in più entità uguali; è possibile inoltre separare un raggruppamento temporaneo in uscita dal batch, nelle entità originali;
- Record: è il processo di raccolta delle statistiche.

o Moduli data: descrivono il sistema staticamente, settano valori e condizioni per l'intero modello e non vengono inseriti nella finestra principale.

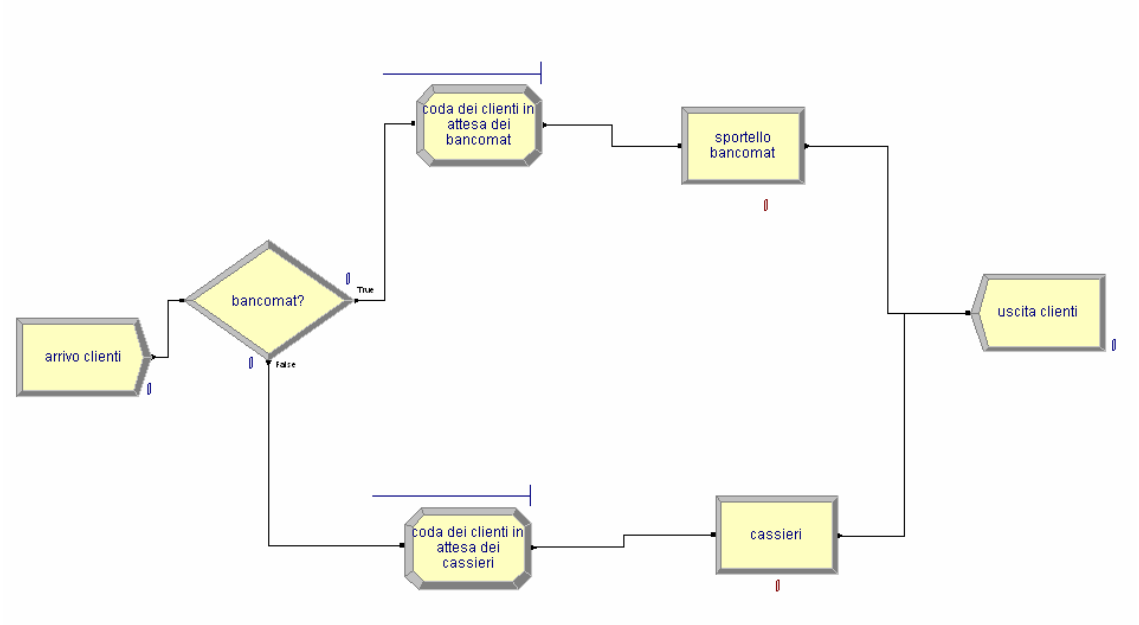


- Attribute: è un modulo data che viene utilizzato per definire il tipo di attributo che ha una specifica entità, le sue dimensioni e il valore iniziale; è possibile assegnare nuovi valori agli attributi grazie al modulo assign;
- Entity: definisce e modifica i vari tipi di entità e la loro immagine raffigurante durante l'esecuzione della simulazione;
- Queue: viene utilizzato per modificare le regole e le caratteristiche di attesa di una specifica coda; la coda si presenta prima di un modulo process e può seguire la logica FIFO o LIFO;
- Resource: definisce le risorse nel sistema di simulazione e comprende anche informazioni sulle disponibilità e sui costi;
- Schedule: utilizzato in collaborazione con il metodo resource, gestisce un programma operativo per una risorsa; utilizzato con il modulo create, definisce un programma di arrivo;
- Variable: definisce le dimensioni e i valori di una variabile.

I vari moduli flowchart e data sono relazionati dai nomi degli oggetti e tutti questi nomi nel modello devono essere univoci, anche i nomi dei moduli.

3.Primo esempio: filiale di una banca

Quresto esempio di modello considera un semplice sistema di servizio di una filiale di una banca, costituita da uno sportello automatico, che offre agli utenti un servizio rapido di self-service per il prelevamento di contanti, e da dei cassieri, che compiono invece operazioni più complesse, come ad esempio il pagamento delle bollette. L'obbiettivo è quello di valutare, a seconda del tasso di arrivo dei clienti, il numero necessario di sportelli automatici e di cassieri affinché le rispettive code non si congestionino.



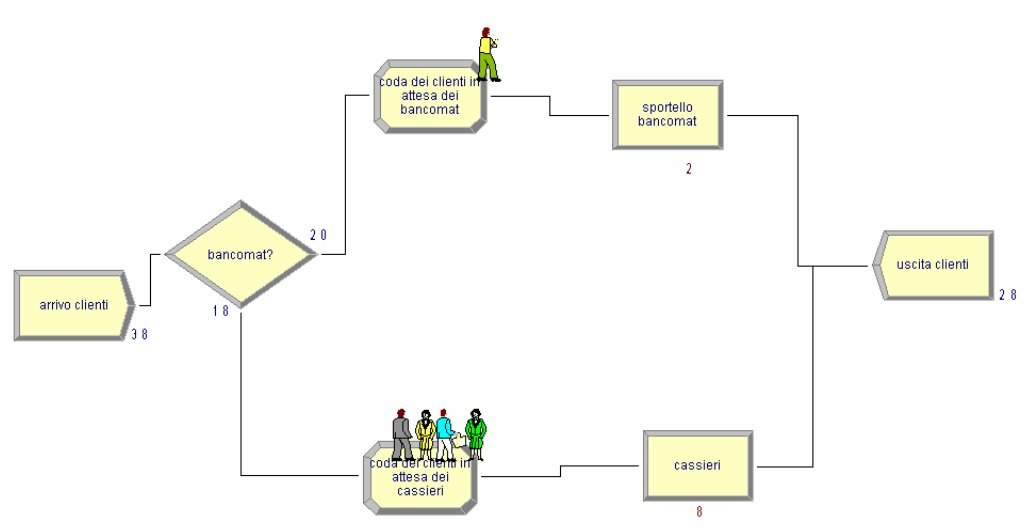
Questo è il modello creato in Arena.

- Il modulo create è stato denominato "arrivo clienti" ed è stata impostata una frequenza di arrivo dei clienti pari a 0,67 clienti/minuto.
  È stato inoltre impostato una distribuzione costante e per semplicità anche che possa arrivare una sola entità contemporaneamente;
- Il modulo decide è stato chiamato "bancomat?" perché qui avviene la decisione sul cliente: chi deve solo fare bancomat, seguirà il flusso del true, cioè la risposta positiva alla decisione; viceversa, chi deve andare dai cassieri, seguirà il flusso del false, cioè la risposta negativa alla domanda; la decisione è a due vie (si o no, oppure, vero o falso) e la probabilità dei clienti di andare da una parte o dall'altra l'abbiamo settata al 50%;
- I due moduli assign sono stati chiamati rispettivamente "coda dei clienti in attesa dei bancomat" e "coda dei clienti in attesa dei cassieri"; al primo è stata assegnata

una capacità massima di 15 clienti, mentre al secondo una quantità massima di 20 clienti; cioè vuol dire che non sono ammissibili code più lunghe di queste, e questo determinerà quindi il numero di cassieri e di bancomat necessari per non far congestionare la coda;

- I due processi “sportello bancomat” e “cassieri”, sono stati scelti come seize delay release, cioè dopo il tempo di processamento i clienti precedentemente allocati vengono rilasciati; il tempo di elaborazione è distribuito in modo triangolare in entrambi i casi: nel bancomat i valori sono medio di 1 minuto, minimo di 0.8 minuti massimo di 1.3 minuti, mentre peri cassieri i tempi che interessano ogni cliente sono con il valore minimo di 2.5 minuti, valore medio di 6 minuti e valore massimo di 11 minuti;
- Il modulo dispose è stato infine chiamato “uscita clienti” ed è la fine del nostro modello.

Se mandiamo in esecuzione il modello e testandolo su un arrivo di 50 clienti, e cambiamo i vari valori di quantità di sportelli e cassieri, notiamo che uno sportello è più che sufficiente e non si creano particolari code, mentre per i cassieri notiamo che un solo cassiere è ampiamente insufficiente perché la coda si allunga molto velocemente; due cassieri già migliorano la situazione però potrebbero esserci delle situazioni in cui la coda si congestiona; con tre cassieri la situazione migliora decisamente e rispettiamo i vincoli di coda imposti, anche se nei momenti dove ci sono più clienti, la coda potrebbe essere superiore alla decina di persone; con 4 cassieri, non ci sono problemi di alcun tipo.
Quindi si potrebbe concludere che il numero ideale di cassieri è tra i 3 e i 4, a seconda dell’affluenza dei clienti.



Dai risultati è possibile verificare che i tempi di attesa delle code dei cassieri sono maggiori dei tempi di attesa dello sportello bancomat:

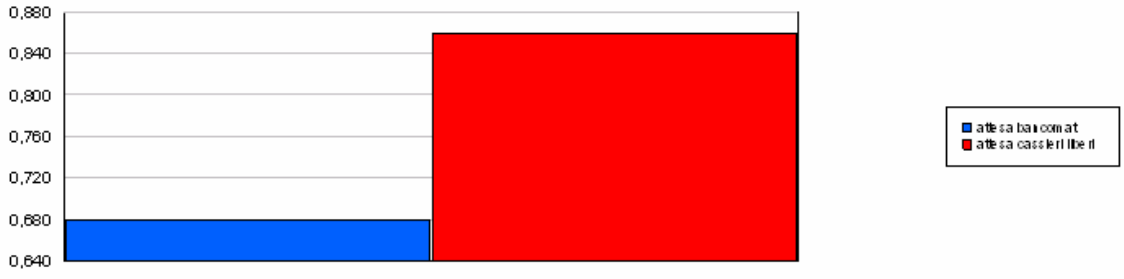
Queue

Time

Waiting Time	Average	Half Width	Minimum Value	Maximum Value
cassieri.Queue	0.04859498	(Insufficient)	0.00	0.1210
sportello bancomat.Queue	0.01067487	(Insufficient)	0.00	0.03775294

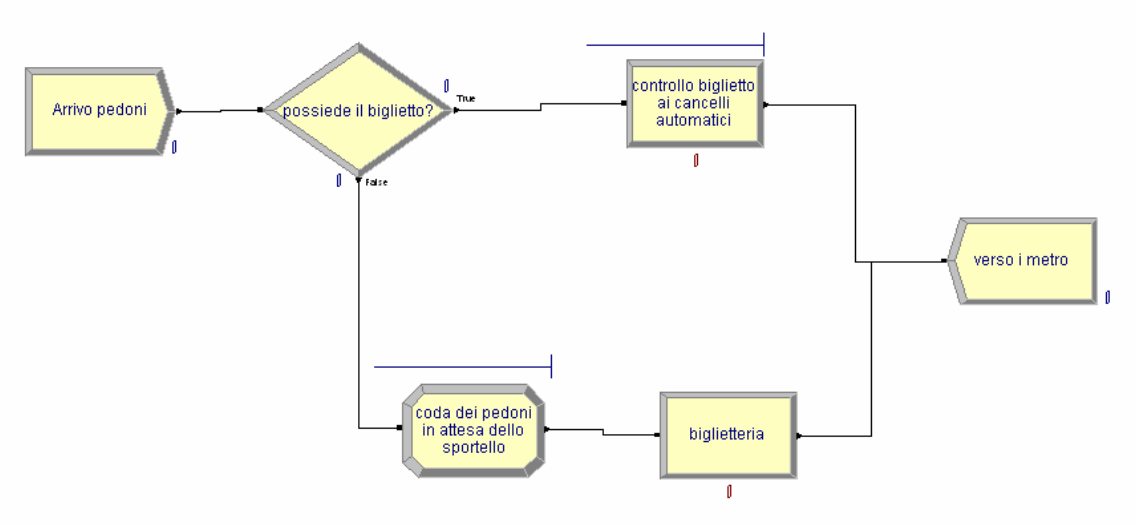
Other

Number Waiting	Average	Half Width	Minimum Value	Maximum Value
cassieri.Queue	1.4514	(Insufficient)	0.00	6.0000
sportello bancomat.Queue	0.4058	(Insufficient)	0.00	3.0000



4.Secondo esempio: stazione di una metropolitana

Questo esempio analizza invece l'ingresso dei passeggeri in una stazione della metropolitana, creando un modello semplice che simuli il flusso di pedoni.
Il fine è quello di stimare l'affluenza massima che può essere gestita affinché l'ingresso non si congestioni, ed eventualmente studiare delle possibili soluzioni alternative.

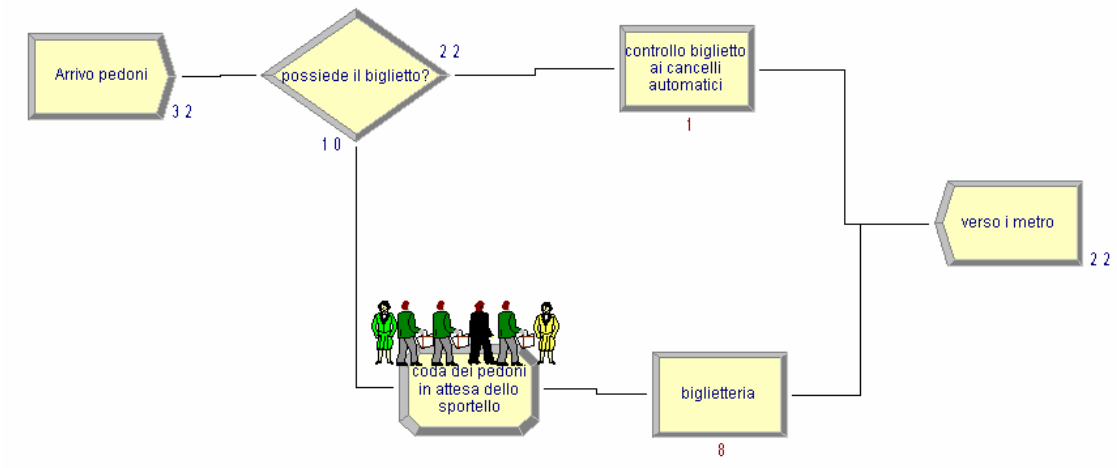


Questo è il modello creato in Arena.

- Il modulo create è stato chiamato "arrivo pedoni", creando l'entità "pedone" e settando la frequenza d'arrivo massimo, costante e a 5000 persone/ora;
- Il modulo decide "possiede il biglietto?" crea la divisione tra i passeggeri tra quelli che hanno già acquistato il biglietto precedentemente e quelli che devono ancora acquistarlo; è una decisione a due vie e la probabilità con cui i pedoni si dividono tra chi ha già il biglietto e chi no è dell'85% per i tornelli e del 15% per le biglietterie. In questo modello si è ipotizzato che il numero di passeggeri che sono già in possesso del biglietto sia molto maggiore rispetto al numero di quelli sprovvisti;
- Il processo "controllo biglietto ai cancelli automatici" serve appunto per controllare che tutti i passeggeri diretti alla metropolitana abbiano i biglietti; è di tipo seize delay release, cioè dopo il tempo di processamento i passeggeri vengono rilasciati; il tempo di elaborazione è distribuito in modo uniforme, compreso tra un valore minimo di 2 secondi e un valore massimo di 3 secondi; il numero di cancelli automatici disponibili è 4;

- Il modulo assign “coda dei pedoni in attesa dello sportello” tiene conto appunto della coda di pedoni che si potrebbe creare alla biglietteria;
- “biglietteria” è il processo che rende disponibile ai passeggeri l’acquisto del biglietto; l’azione è seize delay release, cioè dopo il tempo di processamento i passeggeri vengono rilasciati, di tipo triangolare: il valore minimo è 15 secondi, il valore massimo è 35 secondi, mentre il valore medio è di 25 secondi; è stato programmato che le biglietterie siano 2;
- Infine c’è il modulo “verso i metro”, che conclude il modello e porta i passeggeri alla metro.

Il modello appena creato evidenzia cosa succede all’interno della stazione al variare del flusso di pedoni.
Se mandiamo in esecuzione il modello, infatti, vediamo che per un’affluenza maggiore di 2500 persone all’ora, la stazione si congestiona, mentre non si verificano assolutamente problemi per flussi minori di 1500 persone all’ora.
Un’analisi più accurata mostra che i cancelli automatici non presentano problemi: le code sono snelle e scorrevoli anche nelle ore più critiche.
Quindi, quattro tornelli risultano adeguati per questo modello.
I problemi più gravosi si manifestano invece in prossimità delle biglietterie, dove, considerato un tempo medio di permanenza di 25 secondi a persona, nei momenti di affluenza maggiore, due sportelli risultano insufficienti.
Sarebbe consigliabile, dunque, aumentare il numero di sportelli, migliorando così anche l’efficienza della stazione.



Queue

Time

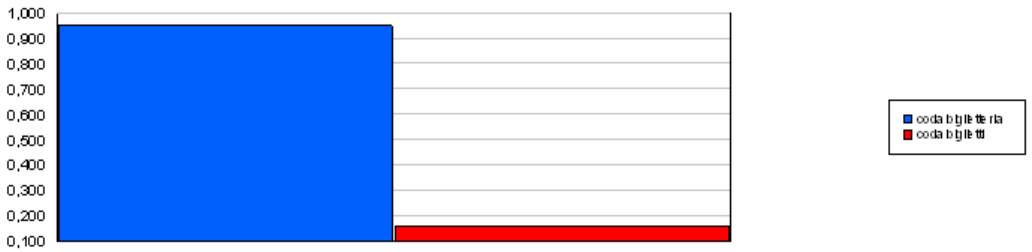
Waiting Time	Average	Half Width	Minimum Value	Maximum Value
biglietteria.Queue	0.02381082	(Insufficient)	0.00	0.04653495
controllo biglietto ai cancelli automatici.Queue	0.00	(Insufficient)	0.00	0.00

Other

Number Waiting	Average	Half Width	Minimum Value	Maximum Value
biglietteria.Queue	6.2554	(Insufficient)	0.00	13.0000
controllo biglietto ai cancelli automatici.Queue	0.00	(Insufficient)	0.00	0.00

Quindi l'affluenza ottimale dei pedoni è stimata tra le 1500 e le 2500 persone/ora, dove le code risultano abbastanza brevi e costanti nel tempo; aumentando la frequenza d'arrivo dei pedoni si riscontrano gravi problemi in prossimità delle biglietterie e questo può essere evitato incrementando il numero di biglietterie, migliorando in questo modo l'efficienza della stazione.

coda biglietteria	0.9496
coda biglietti	0.1600



CAPITOLO 5
Confronto tra i due diversi linguaggi

- Innanzi tutto un differenza di carattere generale: mentre il linguaggio AnyLogic lo troviamo anche in italiano, Arena lo troviamo solo in inglese.
  Questo non è considerato un gran problema perché ormai tutti dovrebbero conoscere ed essere interessati a conoscere l'inglese, ma quando siamo di fronte ad un ambiente tecnico come questo, dove ci sono termini specifici che possono essere non capiti o fraintesi, non è sempre facile trovarsi a proprio agio con una lingua che non si conosce alla perfezione.
  E stiamo parlando di due versione entrambi per gli studenti, perché Arena è nella versione student, mentre AnyLogic è nella versione University 6.
- Per quanto riguarda la parte prettamente grafica, AnyLogic è un passo più avanti di Arena, perché definisce per gli oggetti dei simboli migliori, più intuitivi che ti fanno capire subito che tasto stai scegliendo, invece Arena utilizza sempre gli stessi simboli, ovviamente con nomi diversi e questo non aiuta nella scelta se si è in difficoltà.
  Possiamo dire che Arena è quasi più "formale" di AnyLogic.
  Grazie a questa propensione alla grafica e ai dettagli, AnyLogic è un linguaggio più immediato, perché già solo vedendo i simboli, si intuisce cosa l'utente sta richiedendo al programma, mentre in Arena, per un utente che parte da zero, è necessario e fondamentale leggere la guida.
  Inoltre AnyLogic ha una scelta più felice dei nomi attribuiti ai vari tasti, rendendone anche in questo caso più facile la comprensione.
- Come linguaggio di programmazione invece, AnyLogic è più chiaro e più fluido, e accompagna di più l'utente nelle varie decisioni che deve prendere per costruire il proprio modello.
  È più un linguaggio step by step, che segue l'utente passo passo nella creazione del modello.
  Anche il fatto che ogni oggetto inserito abbia tutte le sue proprietà in unica tabella è utile, perché permette di tenere tutto sotto controllo e di vedere subito se è stato commesso un errore; al contrario invece in Arena è fondamentale richiamare sempre e continuamente altri oggetti che devono essere collegati tra loro per funzionare, come ad esempio i moduli flow-chart e i moduli data: nessuno dei due funziona da solo, ma bisogna complementarli per far funzionare il modello.

- Entrambi possono essere ritenuti linguaggi semplici da imparare e da conoscere e sono ricchi di documentazione nella guida, e questo è importante e si traduce in un vantaggio rispetto agli altri programmi simili perché programmare e utilizzare programmi software non è mai cosa semplice.
- Sebbene entrambe le versioni siano per studenti, Arena tiene comunque un approccio più da esperti, perché oltre ad imparare il linguaggio in sé, è utile avere anche una conoscenza minima di base di informatica per poter capire al meglio le potenzialità del linguaggio.
  Per esempio, lo stesso fatto di dover collegare flow-chart e moduli data non è così immediato, dovuto soprattutto al fatto che dei moduli sono visibili, mentre gli altri non sono visibili ma servono solo per impostare le varie proprietà del modello: il richiamo ad altri oggetti è una operazione prettamente informatica, e un utente che usa programmi simili per la prima volta potrebbe trovarsi un attimo spiazzato e non capirne la logica che ci sta dietro.
- Sempre per venire incontro all'utente, AnyLogic ha inserito nel programma, dei modelli già preparati, che possono essere consultati, una specie di tutorial: fornisce quindi una procedura guidata e comprensibile per la creazione di modelli.
  E non è tutto, perché è possibile anche modificare questi modelli preesistenti in cui l'ambiente è già tutto definitivo, perciò, è possibile creare un nuovo modello non partendo da zero, ma facendosi aiutare per avere una guida ed imparare ad utilizzare il programma iniziando da un semplice modello standard come punto di partenza.
- Arena è considerato un linguaggio più industriale; infatti moltissime aziende utilizzano questo programma per eseguire i loro modelli di simulazione grazie al fatto che con Arena è possibile eseguire modelli molto complicati e complessi.
  Se guardiamo anche al semplice fatto delle immagini che troviamo di default nel programma, vediamo che l'ambiente di lavoro è spinto verso un approccio più industriale che quotidiano, perché vediamo stilizzati forni, viti, caldaie, edifici, cartelli stradali, tubi, macchinari ecc., tutti oggetti che sono poco utili per gli utenti che non hanno esigenze aziendali ma devono costruire un modello su un argomento più comune.
  Un linguaggio è ritenuto lo strumento migliore quando è dipendente dalle necessità e dal tipo di processo da simulare e da come il modello deve essere usato.
  La società di Arena è specializzata sui temi dello sviluppo di modelli con tool differenti e quindi oltre a distribuire Arena utilizzano altri diversi tool in modo da utilizzare sempre il più idoneo allo specifico caso.

- Arena può essere ritenuto più potente come linguaggio di simulazione in sé e lascia più libertà all'utente, mentre AnyLogic come abbiamo detto prima segue di più l'utente, gli da più paletti, più vincoli, quindi certo è più difficile commettere errori e non capire dove sono, ma dall'altra faccia della medaglia questo può limitare la potenza del modello che si vuole eseguire.
  Arena ad oggi è uno dei motori di calcolo più performante dai riscontri statistici, perché è possibile automatizzare con una certa facilità la creazione di modelli di qualsiasi genere e complessità.
- Arena ha l'installato più ampio al mondo, quello più usato in ambito di ricerca.
  Questa grande diffusione ha come conseguenza il fatto che è un programma utilizzato da tanti specialisti in aziende di grandi dimensioni, quindi è necessario per loro acquisire competenze su uno strumento divenuto uno standard di mercato.
  Dietro ad Arena c'e una multinazionale (Rockwell Automation) con elevate capacità di investimento che ne garantisce stabilità ed evoluzioni continue.
  In molte occasioni, inoltre, l'azienda sviluppatrice del software, fornisce corsi di apprendimento e aggiornamento per tutti gli utenti che ne sono interessati.
  Anylogic, invece, pur essendo anch'esso in generale un buon tool, è sicuramente meno diffuso di Arena perchè meno performante su modelli complessi.
- Un punto a favore di AnyLogic è il fatto che sia implementata una funzione per l'utilizzo di Java, che è forse il linguaggio di programmazione orientato ad oggetti più diffuso al momento.
  Questo permette di essere un linguaggio completo sia perché è possibile eseguire vere e proprie animazioni interattive del modello ricorrendo appunto alla programmazione orientata ad oggetti, sia perché permette ad AnyLogic di potersi adattare a qualsiasi sistema operativo, dato che Java è un linguaggio diffuso anche sul web quindi tutti i sistemi operativi possono essere in grado di codificarlo.
  Il problema di questa implementazione è che è necessaria una buona conoscenza di base di Java per poter utilizzare al meglio AnyLogic perché i suoi richiami sono numerosi.
  Arena invece non utilizza il linguaggio Java, ma utilizza il linguaggio Visual Basic (VBA), che è un linguaggio che non permette una vera e propria animazione, ma solo un'importazione di immagini dinamiche per comprendere al meglio il modello, quindi l'animazione può essere sviluppata solo con moduli appositi e si compone di immagini stilizzate già importate di default in Arena, come ad esempio macchinari, pulsanti, cataloghi, disegni industriali, ecc.

In questo caso però, conoscere il linguaggio VBA non è strettamente necessario conoscerlo in quanto viene proposto all'interno di Arena in quantità non enorme; si può utilizzare benissimo Arena anche senza conoscere il linguaggio Visual Basic. Inoltre Arena è un'applicazione solo di Windows, quindi non è possibile interfacciarsi con tutti gli ambienti come invece riesce a fare AnyLogic.

- Java in AnyLogic permette al linguaggio anche di importare una grande quantità di dati esterni, come ad esempio piantine di edifici che possono essere utilizzati come base di partenza del proprio modello e che costituiscono il layout dell'ambiente di lavoro.
  Oppure ci sono vere e proprie immagini che danno al modello quel tocco in più; per esempio sono già importate di default immagini di medici in camice, oppure di pazienti, e questo è utile sia per migliorare la comprensione del modello, sia per renderlo migliore all'occhio di chi lo vede in esecuzione.
- Un punto debole di Arena è invece il 3D, che però è completamente rinnovato nella prossima versione in uscita e la quale sarà implementata senza aggiunta di costi in tutte le versioni acquistabili dai clienti.

- In ogni caso, in generale, per determinare quale strumento è migliore, bisogna analizzare caso per caso perché ogni programma ha dei pro e dei contro che possono manifestarsi in un modello piuttosto che in un altro.
  Sarebbe quindi utile conoscere e poter disporre di più strumenti di simulazione per poter scegliere ad ogni situazione quale conviene utilizzare per il proprio modello.

CONCLUSIONI

In questa tesi ci si è prefissati l'obiettivo di confrontare due modelli su due programmi di simulazione differenti, per poterli valutare e confrontare.
I due modelli presi in esame riguardano uno, l'analisi di una filiale di una banca, con la gestione dei clienti in arrivo dividendoli tra chi doveva effettuare un'operazione self-service al bancomat e chi invece aveva bisogno di operazioni alle casse; e il secondo, l'analisi di una stazione di una metropolitana, dividendo il flusso di passeggeri in base a chi aveva già effettuato l'acquisto del biglietto e chi invece doveva recarsi alla biglietteria prima di accedere ai cancelli automatici della metropolitana.
Come struttura dell'elaborato, si è prima spiegato un po' l'argomento generale, con nozioni teoriche sulle parole modello e simulazione, poi si è passati ad una veloce introduzione e spiegazione dei due linguaggi utilizzati per il confronto, e si è descritto per ciascuno il modello e il suo svolgimento, e si sono fatte le dovute conclusioni.
Infine si è fatto un confronto critico tra i due programmi spiegando vantaggi dell'uno rispetto all'altro e viceversa, ambiti di esecuzione, aspetti da migliorare ecc.

BIBLIOGRAFIA

[1] Wikipedia (http://it.wikipedia.org/wiki/).

[2] AnyLogic (http://www.xjtek.com/anylogic/why_anylogic/).

[3] Rockwell Automation (http://www.rockwellautomation.com/).

[4] Arena Simulation Software (http://www.arenasimulation.com/).